对象的职责就是决定如何对这条消息作出反应（执行相应的代码）;

New出来的对象位于堆中，但基本类型位于栈中；

Static方法的重要用法就是不创建任何对象的前提下调用。

Super: super可以理解为是指向自己超（父）类对象的一个指针，而这个超类指的是离自己最近的一个父类。

super也有三种用法：

1.普通的直接引用

与this类似，super相当于指向当前对象的父类，这样就可以用super.xxx来引用父类的成员。

2.子类中的成员变量或方法与父类中的成员变量或方法同名:super指调用父类的方法或变量。

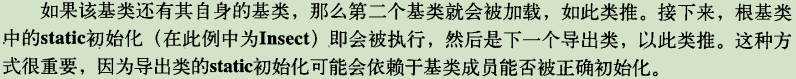
3.引用构造函数

super（参数）：调用父类中的某一个构造函数（应该为构造函数中的第一条语句）。

this（参数）：调用本类中另一种形式的构造函数（应该为构造函数中的第一条语句）。

调用子类构造方法是会先调用父类构造方法（只适用于默认构造器）；对于有参数的构造器就要用super（i）显示地调用分类构造器。

类的代码在初次使用时才加载。当访问static域或是static方法时也会被加载。



多态从另一方面将接口和实现分离 。

绑定：把一个方法与其所在的类/对象 关联起来叫做方法的绑定。绑定分为静态绑定（前期绑定）和动态绑定（后期绑定）。

静态绑定（前期绑定）是指：在程序运行前就已经知道方法是属于那个类的，在编译的时候就可以连接到类的中，定位到这个方法。在Java中，final、private、static修饰的方法以及构造函数都是静态绑定的，不需程序运行，不需具体的实例对象就可以知道这个方法的具体内容。

动态绑定动态绑定（后期绑定）是指：在程序运行过程中，根据具体的实例对象才能具体确定是哪个方法。典型例子就是父类引用指向子类对象。

静态方法无法被重写，

父类构造方法中调用子类方法，可以调用，但子类此时尚未初始化完成，调用的属性并非是赋的值，而是默认值；

放在接口中的任何类都自动的是public和static的

Stack栈，先进后出。由linkedList提供支持。

Queue队列，先进先出。由linkedList提供支持。

String 对象时不可变的，每一个修改方法实际上都是创建一个全新的String对象，而最初的String对象纹丝未动。而使用StringBuilder则只生成一个StringBuilder对象，它还允许预先为其指定大小，避免多次重新分配缓冲。

Scanner可以用useDelimiter（）来设置定界符（即符号分隔文件）。

Java程序在他开始运行之前并不是将所有类都完全加载，而是在必需的时候才加载。Java属于动态加载语言，c++则属于静态加载语言。

Java在泛型代码内部是无法获得任何有关泛型参数类型的信息，泛型是通过擦除来实现的，意味着使用泛型是所有的类型信息都被擦除了，唯一想知道的就是你在使用一个对象。List<String>和List<Interger>其实是同一个类型，都最终被擦除成List。类型只有在静态类型检测时才出现，在此之后都将被擦除。擦除的目的是为了使用非泛化的类库，在不破坏现有类库的情况下将泛型融入到java语言。

数组和其他种类容器之间的区别：效率，类型和保存基本类型的能力。在java中数组是一种效率最高的存储和随机访问对象引用序列的方式。数组就是一个简单的线性序列，使得元素访问非常迅速。代价就是数组对象的大小别固定且在其生命周期里不可改变。

Map的entrySet中的map.entry对象都只存储了它的索引，而不是实际的键值，当你调用getKey（）和getValue（）时，他们就会使用该索引来返回恰当的data元素。

Map中使用的键，任何键都必须具有一个equals（）方法，如果键被用于散列map，那么他必须具有恰当的hashCode（）方法；如果键被用于TreeMap,那么他必须实现Comparable。

散列是映射存储元素时常用的方式，数组并不保存键本身，而是通过键对象生成一个数字，将其作为数组的下标，这个数字就是散列码。

hashSet（）基于hashMap（）；treeSet（）基于TreeMap()；

问题：接口的优势理解不深，内部类优势理解不深, 对于类的理解不深（类型信息）